agents & threads

In a previous collection of exercises you investigated threads in Java by building a series of small programs to bounce concurrent "*blobs*" around a JFrame, using some (independent) mechanism to monitor their positions & cause them to bounce.

The purpose of this practical session is to re-engineer your earlier work using agents to investigate patterns of concurrent collaborative activity.

advice

- think of the JFrame/Graphics panel as the "world". The world is a global singleton so specify a non-cloneable agent to handle requests to make changes to the world;
- think about the nature of the monitoring process, this should also be specified as some kind of agent-ware (at some stage consider whether/how this process could/should interface with the world);
- remember that messages are sent in the current thread but this thread does not wait for messages to be processed before continuing;
- remember also that there can be no guarantee that (i) messages do not change order in transit (ii) that all agents run at the same speed (some may over-run and/or under-run);
- blobs will still need to be threaded in some way (either using Java Threads or Boris agent timers).

NB: sometimes we provide a skeletal structure to help you get started (check the web-page). This has some classes provided for you & a couple of agents partially set up but do not assume that the incomplete messaging/collaboration patterns are suitably/sensibly structured.

stages

1. start with one blob, the world and some kind of controller/monitor to advise the blob when to bounce. Initially bounce off the sides of the JFrame (or some region within it). Communication between blob, world & controller should be by agent messaging;
2. add some additional *blobs* (they should move independently – each running in their own thread). The controller should detect collisions between the blobs. A collision is when the blobs (or, for simplicity, the squares they are drawn in) overlap. When there is a collision, the controller messages the blobs to cause them to "bounce" away from the collision
3. extend the program so it can handle multiple (ie: lots more than 2) blobs. Examine it carefully to check for synchronisation glitches and threading problems, agent over-run & under-run;
4. sketch out a diagram showing the collaboration patterns between your agents. Think about alternatives, advantages, disadvantages, etc.